มุม
ภาษา C

การเขียนโปรแกรมแบบเรสซิเดนต์เบื้องต้น

■ จเร เลิศสุดวิชัย

ปัจจุบันภาษาซี เริ่มเข้ามามีบทบาทมากขึ้นต่อการพัฒนาซอฟต์แวร์ต่าง ๆ อย่างเช่น โลตัสเวอร์ชัน 3 ซึ่งเป็นเวอร์ชันใหม่ก็ถูกพัฒนาด้วยภาษาซี เนื่องจากภาษาซีเป็นภาษาโครงสร้าง (Structure program) ซึ่งสามารถแบ่งแยกโปรแกรมออกเป็นโมดูล การแก้ไขข้อผิดพลาดการทำงานของโปรแกรมจึงทำได้ง่าย นอกจากนั้นภาษาซียังเป็นภาษาระดับสูงที่มีการทำงานได้รวดเร็ว ตัวแปลภาษาซี (C compiler) หลายตัวในปัจจุบันมีความสามารถในการแปลให้ได้โปรแกรมภาษาเครื่อง (Machine code) ที่สั้น ทำให้ได้โปรแกรมที่ทำงานได้รวดเร็ว และที่สำคัญที่สุดคือภาษาซีเป็นภาษาที่สามารถนำไปทำงานได้หลายระบบ (Portable language) การเขียนโปรแกรมภาษาซีบนเครื่องคอมพิวเตอร์เครื่องหนึ่งสามารถที่นำโปรแกรมเดิมนี้ไปทำงาน (Run) บนเครื่องคอมพิวเตอร์อีกเครื่องหนึ่งได้ โดยอาจจะไม่ต้องเปลี่ยนแปลงหรือเปลี่ยนแปลงโปรแกรมบางส่วนเพียงเล็กน้อย (ขึ้นอยู่กับลักษณะการเขียนโปรแกรมและการใช้ฟังก์ชันที่ตรงกับมาตรฐานมากน้อยเพียงใด) โดยนำโปรแกรมมาแปลใหม่ด้วยตัวแปลภาษา (Compiler) บนเครื่องคอมพิวเตอร์นั้น และสุดท้ายลักษณะรูปแบบโปรแกรมภาษาซีมีความอ่อนตัว (Flexible) คือ สามารถจะเขียนโปรแกรมได้หลายรูปแบบในการทำงานแบบเดียวกัน ลักษณะนี้แม้จะเป็นส่วนดีสำหรับนักเขียนโปรแกรมภาษาซีที่มีความชำนาญ แต่สำหรับผู้เริ่มจะเรียนรู้ภาษาซีใหม่ ๆ จะกลายเป็นการสร้างความสับสนต่อการเขียนโปรแกรมในระยะเริ่มต้น

เปิดมุม

จุดประสงค์ของมุมภาษา C นี้จะเป็นการเสนอเทคนิคต่าง ๆ ในการเขียนโปรแกรมภาษาซี และตอบปัญหาสำหรับท่านผู้อ่านที่เขียนโปรแกรมด้วยภาษาซีอยู่แล้วพบปัญหาที่ไม่สามารถจะแก้ไขได้ โดยบทความในมุมภาษา C แต่ละตอนจะแบ่งออกเป็น 2 ส่วนด้วยกันคือ ส่วนแรกจะเป็นส่วนของการตอบปัญหาที่พบในการเขียนโปรแกรม ท่านผู้อ่านที่พบปัญหาสามารถส่งคำถามมาได้โดยตรงที่วารสารไมโครฯ และวงเล็บที่มุมซองว่ามุมภาษา C ในส่วนที่สองนั้นจะเป็นส่วนของการเสนอเทคนิคโปรแกรมภาษาซี โปรแกรมและเทคนิคต่าง ๆ นั้น ช่วงระยะนี้คงจะอยู่บน MS-DOS หรือ PC-DOS และโปรแกรมแปลภาษาก็คงจะเป็นเทอร์โบซี ไมโครซอฟต์ซี หรือตัวแปลภาษาซีตัวอื่นที่มีใช้กัน เทคนิคและโปรแกรมที่จะนำมาลง ถ้าท่านผู้อ่านท่านใดที่มีความสนใจเป็นพิเศษในเรื่องใดก็เสนอแนะได้ครับ สำหรับเทคนิคโปรแกรมในฉบับนี้จะเป็นเรื่อง "การเขียนโปรแกรมแบบเรสซิเดนต์ด้วยภาษาซี" ซึ่งจะมีตัวอย่างโปรแกรมของเทอร์โบซี และไมโครซอฟต์ซีเป็นตอนที่หนึ่ง ส่วนตอนที่สองในฉบับถัดไปจะเพิ่มเทคนิคโปรแกรมแบบเรสซิเดนต์ที่เป็นป็อบอัพ (Pop-up) คล้ายโปรแกรม Side Kick

ปัญหาภาษาซี

ปัญหาของมุมภาษา C ได้ตัดมาจากจดหมายถึงไมโครฯ ซึ่งความจริงจะต้องตอบในส่วนของจดหมายถึงไมโครฯ แต่เห็นว่าเป็นปัญหาของการเขียนโปรแกรมภาษาซี จึงขออนุญาตนำมาตอบในมุมภาษา C นี้แทน

การดัดแปลงโปรแกรมจากภาษาปาสคาลเป็นภาษาซี

คุณสุเทพ รัตนธารเกียรติ

ถาม ผมมีปัญหาเกี่ยวกับภาษาซี ซึ่งผมได้ดัดแปลงโปรแกรมจากภาษาปาสคาลมาเป็นภาษาซีตามโปรแกรมในรูปที่ 1

การดัดแปลงโปรแกรมของผมแปลงจากภาษาปาสคาลเป็นภาษาซีบรรทัดต่อบรรทัด แต่ปรากฏเมื่อคอมพายล์ด้วยเทอร์โบซี (Turbo C) และควิชค์ซี (Quick C) ผ่านได้ไม่เกิดข้อผิดพลาด แต่เมื่อนำไปใช้งานจะทำให้เครื่องแฮงทันที ประโยคที่คิดว่าผิดคงเป็น

```
a[n+1]=7 ;
```

แน่นอน แต่สงสัยว่าทำไมในภาษาปาสคาล (ใช้เทอร์โบปาสคาล) จึงทำงานถูกต้องไม่มีปัญหา

```
main()
{
  int i,n;
  int a[5];

  for(i = 0;i<5;i++)
    a[i] = i;
  i = 0;  n = 5;  a[n+1] = 7;
  while(7 != a[i])
    i++;
  if(i == n+1)
    printf("not found 3");
  else
    printf("found 3");
}
```

รูปที่ 1

ตอบ จากโปรแกรมที่ให้มานั้น คุณสุเทพหาที่ผิดได้ถูกแล้ว การอ้างถึงข้อมูลที่เป็นอะเรย์ในภาษาซีแตกต่างกับภาษาอื่น คือ ตัวชี้ (INDEX) จะเริ่มที่ศูนย์จนถึงค่ามากที่สุดในการจองอะเรย์ลบด้วยหนึ่ง ดังในโปรแกรมที่ให้มา ซึ่งจองตัวแปร a เป็นข้อมูลชนิดอะเรย์ของจำนวนเต็ม 5 ข้อมูล

int a[5]; → | a[0] | a[1] | a[2] | a[3] | a[4] |

การอ้างถึงข้อมูล a ตัวแรกตัวชี้จะต้องเป็นศูนย์ คือ a[0] ส่วนข้อมูล a ตัวสุดท้ายคือ a ตัวที่ 5 ตัวชี้จะเป็น 4 คือ a[4] ในโปรแกรมบรรทัดที่มีข้อความว่า a[n+1] = 7 ตัวชี้จะเป็น 5+1=6 ซึ่งจะเห็นว่าการอ้างตัวแปรตัวชี้เป็น 6 นั้นเกินพื้นที่หน่วยความจำที่เราจองเอาไว้ อาจเขียนค่า 7 ทับลงในส่วนอื่นของโปรแกรม ทำให้เครื่องเกิดการแฮงได้ ดูจากลักษณะโปรแกรมที่คุณสุเทพให้มายังมองไม่ออกว่าโปรแกรมนี้ทำงานอะไร

ขนาดของไฟล์จากการก๊อบปี้ไฟล์

คุณสมชาย นิลน้อย

ถาม ผมเพิ่งเริ่มเขียนโปรแกรมด้วยภาษาซีไม่นานนัก ได้ลองเขียนโปรแกรมตามตัวอย่างในหนังสือที่ผมอ่านอยู่แล้วทดลองทำงานดู ตามโปรแกรมที่ผมให้มาในรูปที่ 2 เป็นโปรแกรมก๊อบปี้ไฟล์จากไฟล์ที่มีอยู่แล้วไปเป็นไฟล์ใหม่ ปรากฏว่าไฟล์ใหม่ที่ได้จะมีขนาดเท่ากันกับไฟล์เดิม แต่ในโปรแกรมของผมมีตัวนับจำนวนไบต์ของไฟล์ซึ่งนับได้ไม่เท่ากับขนาดไฟล์จริง ๆ ผมใช้เทอร์โบซีเวอร์ชัน 1.0 หาดูเรื่องนี้ในคู่มือก็ไม่พบ ขอให้ทางไมโครฯ ช่วยตอบปัญหาด้วยครับ และช่วยอธิบายวิธีการเปิดไฟล์ในภาษาซีให้ละเอียด เพื่อผมจะได้นำไปใช้งานได้อย่างถูกต้อง

ตอบ เรื่องไฟล์ในภาษาซีของตัวแปลภาษาบน MS-DOS เช่น เทอร์โบซี ไมโครซอฟต์ซี แตกต่างกับไฟล์ในภาษาซีบน UNIX และสร้างความสับสนให้กับผู้เริ่มใช้ภาษาซีมากทีเดียว การเปิดไฟล์ด้วยฟังก์ชัน fopen นั้น รูปแบบการใช้ คือ

```
FILE *fopen (char *filename, char *type);
```

ซึ่ง type จะเป็นการบอกถึงการเปิดไฟล์ขึ้นมาใช้งาน ดังนี้

"r" : เปิดไฟล์เพื่อที่จะอ่าน

"w" : เปิดไฟล์ใหม่เพื่อที่จะเขียน และไฟล์เดิมนั้นมีข้อมูลก็จะถูกเขียนทับ

"a" : เปิดไฟล์เพื่อจะเขียนต่อท้ายไฟล์ แต่ยังไม่เคยมีไฟล์นั้นอยู่ จะสร้างไฟล์ใหม่

"r+" : เปิดไฟล์เพื่อที่จะอ่านและเขียนเพิ่ม

"w+" : เปิดไฟล์ใหม่เพื่อจะอ่านและเขียน ถ้าไฟล์นั้นมีอยู่ข้อมูลจะถูกเขียนทับ

"a+" : เปิดไฟล์เพื่ออ่านและเขียนต่อท้ายไฟล์ ถ้ายังไม่เคยมีไฟล์นั้นอยู่จะสร้างไฟล์ใหม่ให้

สำหรับในเทอร์โบซีและไมโครซอฟต์ซี มีวิธีการปฏิบัติงานเพิ่มขึ้นมาอีก (ซึ่งบน UNIX ไม่มี) คือ t และ b ต่อท้าย type เช่น "rt", "w+t" หรือ "wt+" เป็นต้น โดย t และ b มีความหมายดังนี้

t : เป็นการเปิดไฟล์ในโหมด Text ซึ่งในโหมดนี้ มีข้อพิเศษอยู่คือ ในการอ่านไฟล์นั้นรหัสควบคุม Carriage return และ Line feed (CR-LF) จะถูกรวมเหลือเพียง Line feed (LF) ส่วนในการเขียนไฟล์รหัสควบคุม Line-feed (LF) ก็จะถูกเปลี่ยนเป็นสองรหัสคือ CR และ LF และที่สำคัญคือ รหัส CTRL+Z จะถูกมองเป็นรหัสบอกจบไฟล์

b : เป็นการเปิดไฟล์ในโหมด Binary ซึ่งการอ่านเขียนไฟล์จะไม่มีการเปลี่ยนรหัสเหมือนในโหมด Text

จากโปรแกรมที่ให้มา การเปิดไฟล์คือ

```
in = fopen (argv[1], "r");
```

```
#include <stdio.h>

main (int argc,char * argv[])
{
   FILE  *in, *out;
   long  n;
   int   c;

   if (argc < 3)
       {
       printf ("USAGE : copyx input-file
       out-file \n");
       exit (1);
       }
   in = fopen (argv[1],"r");
  out = fopen (argv[2],"w");
  if ( (in == NULL) || (out == NULL) )
     {
     printf ("Can't open file ");
     exit (1);
     }
  n = 0l;
  while ((c = getc (in)) != EOF)
     {
     putc (c, out);
     n++;
     }
       fclose (in);
  fclose (out);
  printf (" %ld characters copied",n);
}
```

รูปที่ 2

out = fopen (argv[2], "w");

ไม่ได้บอกโหมดในการเปิดไฟล์ ดังนั้นโหมด Default จะถูกกำหนดขึ้นเองตามค่าที่มีอยู่ในตัวแปรของระบบตัวแปลภาษา (ทั้งเทอร์โบซี และ ไมโครซอฟต์ซี) ชื่อ _fmode ในกรณีนี้เป็นไปได้ที่ค่าในตัวแปร _fmode มีค่าเป็นโหมด Text ดังนั้นการอ่านไฟล์รหัส CR และ LF ก็จะถูกเห็นเป็นเพียง LF ทำให้การนับจำนวนไบต์การก๊อบปี้ไฟล์น้อยจากขนาดของไฟล์จริง ๆ แต่ว่าการเขียน LF ลงไฟล์จะถูกเปลี่ยนเป็น CR และ LF ดังนั้น ไฟล์ใหม่ที่ถูกก๊อบปี้จึงเหมือนเดิมทุกประการ ในกรณีของคุณสมชายคิดว่าคงนำโปรแกรมไปใช้ก๊อบปี้กับไฟล์ที่เป็นโหมด Text ถ้านำไปใช้กับไฟล์ที่ไม่ใช่ Text เช่น ไฟล์ .COM , .BIN, .EXE เป็นต้น อาจจะได้ไฟล์ที่มีขนาดเล็กกว่าไฟล์ต้นฉบับก็ได้ เนื่องจากรหัส CTRL+Z อาจอยู่ในส่วนใดของไฟล์พวกนี้ ทำให้การอ่านคิดว่าจบไฟล์แล้วทั้งที่ความจริงยังไม่หมดไฟล์

ดังนั้น เพื่อความถูกต้องในการเปิดไฟล์จริง ๆ ควรจะเพิ่มโหมดในของการเปิดไฟล์ให้เป็นโหมด Binary คือ

in = fopen (argv[1], "rb");

out = fopen (argv[2], "wb");

```
#include <stdio.h>

main (int argc,char * argv[])
{
   FILE  *in, *out;
   long  n;
   int   c;

   if (argc < 3)
      {
      printf ("USAGE : copy1 input-file
        out-file \n");
      exit (1);
      }
   in = fopen (argv[1],"rb");
   out = fopen (argv[2],"wb");
   if ( (in == NULL) || (out == NULL) )
      {
      printf ("Can't open file ");
      exit (1);
      }
   n = 01;
   while (fscanf(in,"%c",&c) != EOF)
      {
      fprintf (out,"%c",c);
      n++;
      }
   fclose (in);
   fclose (out);
   printf (" %ld characters copied",n);
}
```

รูปที่ 3

การอ่านและเขียนไฟล์ด้วยฟังก์ชัน fopen นอกจากจะใช้ getc และ putc แล้วยังสามารถใช้ฟังก์ชัน fscanf และ fprintf ได้ ซึ่งฟังก์ชันทั้งสองสามารถจะกำหนดรูปแบบของการอ่านและเขียนไฟล์ได้ ตัวอย่างของการก๊อบปี้ไฟล์เพียงแต่เปลี่ยน

getc (in,c);

เป็น fscanf (in, "%c", &c);

putc (in,c);

เป็น fprintf (in, "%c", &c);

```
#include <fcntl.h>
#include <sys\types.h>
#include <sys\stat.h>
#include <io.h>
#include <stdio.h>

#define  BUFFER_SIZE 1024
main (int argc,char * argv[])
{
   int  in, out, nr;
   long  n;
   char  buffers[BUFFER_SIZE];

   if (argc < 3)
      {
      printf ("USAGE : copy2 input-file
        out-file \n");
      exit (1);
      }
   in  = open (argv[1],O_RDONLY | O_BINARY);
   out = open (argv[2],O_CREAT | O_TRUNC
        |O_BINARY);
   if ( (in == -1) || (out == -1) )
      {
      printf ("Can't open file ");
      exit (1);
      }
   n = nr = read (in,buffers,BUFFER_SIZE);
   while (nr > 0)
      {
      write (out,buffers,nr);
      nr = read (in,buffers,BUFFER_SIZE);
      n  = n + nr;
      }
   close (in);
   close (out);
   printf (" %ld characters copied",n);
}
```

รูปที่ 4

การเปิดไฟล์	fopen	open หรือ creat
อ่านข้อมูลจากไฟล์	getc หรือ fscanf	read
เขียนข้อมูลลงไฟล์	putc หรือ printf	write
การปิดไฟล์	fclose	close

รูปที่ 5 แสดงการสรุปฟังก์ชันในการใช้งานไฟล์

นอกจากการเปิดปิดด้วยฟังก์ชัน fopen และ fclose แล้วยังมีฟังก์ชันระดับต่ำที่เรียกใช้บริการของโอเอสโดยตรง คือ ฟังก์ชัน create หรือ open และ close ส่วนฟังก์ชันการอ่านและเขียนจะใช้ฟังก์ชัน read และ write ซึ่งตัวอย่างโปรแกรมดังแสดงในรูปที่ 4 และตารางสรุปวิธีการเปิดปิดไฟล์ ฟังก์ชันให้อ่านและเขียนไฟล์ในรูปที่ 5 การอ่านและเขียนโดยใช้ read และ write สามารถทำได้หลายไบต์ในหนึ่งครั้ง ความเร็วในการทำงานจึงเร็วกว่าฟังก์ชัน getc, putc และ fscanf, fprintf

เทคนิคโปรแกรม

ในส่วนเทคนิคโปรแกรมของฉบับนี้ จะกล่าวถึงเรื่องของการเขียนโปรแกรมแบบเรสซิเดนต์ โดยแนะนำหลักการเบื้องต้นพร้อมทั้งตัวอย่างโปรแกรมประกอบ เพื่อนำความรู้ไปใช้ในตอนต่อไปของฉบับหน้า ซึ่งเป็นโปรแกรมเรสซิเดนต์แบบป๊อบอัพ (Pop-up) โดยสามารถกำหนดแป้นพิมพ์ในการเรียกโปรแกรมเรสซิเดนต์ให้ขึ้นมาทำงานได้ หลักการและเทคนิคโปรแกรมเรสซิเดนต์นั้นความจริงถูกเก็บเป็นความลับของผู้พัฒนาโปรแกรมพวกนี้ ในโปรแกรมมีการเรียกใช้ฟังก์ชันของดอส ซึ่งเป็นฟังก์ชันที่ไม่มีบอกในคู่มือการใช้งานของดอส ในส่วนของเทคนิคโปรแกรมนี้ได้พยายามรวบรวมจากหนังสือหลายเล่ม นำหลักการเหล่านี้มาให้ผู้อ่านได้รับรู้ เพื่อให้ผู้อ่านได้นำไปใช้ประโยชน์ต่อไป

โปรแกรมเรสซิเดนต์ คืออะไร

MS-DOS เป็นโอเอสที่ถูกออกแบบมาในลักษณะของ Single Task คือ ทำงานกับโปรแกรมเพียงหนึ่งโปรแกรมในเวลาหนึ่ง นั่นคือในการที่จะเรียกใช้งานโปรแกรมใดโปรแกรมหนึ่งบน MS-DOS จะต้องโหลดโปรแกรมนั้นจากดิสค์ลงมาในหน่วยความจำก่อนและการที่จะใช้งานโปรแกรมอื่นนั้นจะต้องรอจนกว่าโปรแกรมเดิมที่โหลดลงหน่วยความจำนั้นทำงานเสร็จสิ้นเสียก่อน ดังนั้นทำอย่างไรเราจึงจะสามารถทำงานหลายอย่างได้ หรือเรียกใช้โปรแกรมอื่นในขณะที่ทำงานอีกโปรแกรมหนึ่งอยู่ วิธีทำก็คือจะต้องโหลดเอาโปรแกรมที่ต้องการเรียกใช้งานนั้นฝังหรือค้างอยู่ในหน่วยความจำ (Stay-Resident) ก่อนที่จะโหลดเอาโปรแกรมอื่นลงไปเพื่อทำงาน เนื่องจากโปรแกรมยังคงค้างอยู่ในหน่วยความจำ จึงทำให้โปรแกรมพร้อมที่จะถูกเรียกใช้งานได้ตลอดเวลา ซึ่งลักษณะโปรแกรมพวกนี้เราเรียกว่า โปรแกรมเรสซิเดนต์

การฝังโปรแกรมลงหน่วยความจำ

ในการทำงานโปรแกรมตามปกตินั้น โปรแกรมจะถูกอ่านจากดิสค์ลงในหน่วยความจำที่ดอสจัดให้เรียกว่า Transient Program Area และหลังจากจบการทำงานโปรแกรมหน่วยความจำส่วนนี้จะกลับเป็นที่ว่างอีกเช่นเดิม การที่จะฝังโปรแกรมลงในหน่วยความจำนั้น ดอสเองเตรียมบริการไว้ให้ 2 วิธีด้วยกันคือ

- อินเตอร์รัพต์ 27H

DX = Offset ของไบต์สุดท้ายของโปรแกรมบวกด้วย 1 (อ้างอิงจาก Program Segment Prefix)

CS = Segment ของ Program Segment Prefix

การฝังโดยวิธีนี้ จะได้ขนาดสูงสุด 64 กิโลไบต์เท่านั้น

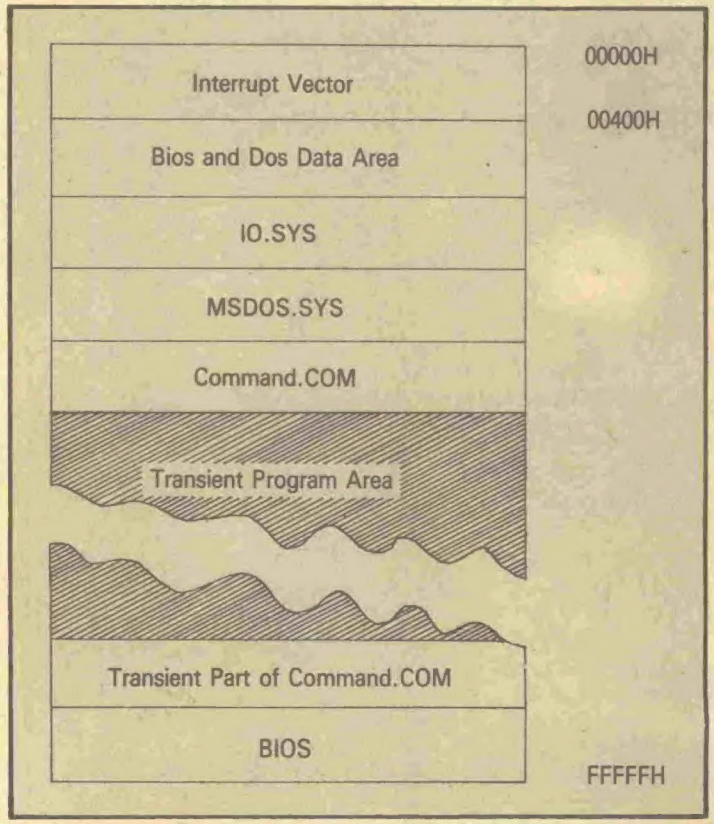
- อินเตอร์รัพต์ 21H ฟังก์ชันบริการ 31H

AH = 31h

AL = Exit Code

DX = ขนาดของโปรแกรมที่จะฝังในหน่วยความจำเป็นพารากราฟ (คือจำนวนไบต์หารด้วย 16)

ตามปกติแล้วโปรแกรมเรสซิเดนต์มักจะเขียนด้วยภาษาแอสเซมบลี ซึ่งจะได้โปรแกรมที่มีขนาดเล็ก เพื่อที่จะได้มีหน่วยความจำเหลือสำหรับใช้งานอื่นต่อไป แต่การที่จะเขียนด้วยภาษาซีหรือปาสคาลก็สามารถทำได้ แต่จะได้โปรแกรมที่มีขนาดใหญ่ขึ้นมาประมาณ 8 กิโลไบต์ สำหรับข้อดีของการเขียนโปรแกรม



รูปที่ 6 แสดงหน่วยความจำหลักโหลดโปรแกรมเพื่อจะทำงาน

เรสซิเดนต์ด้วยภาษาซีหรือปาสคาล นั่นคือ ทำให้เขียนโปรแกรมได้เข้าใจง่าย การฝังโปรแกรมด้วยวิธีทั้งสองวิธีที่กล่าวมานั้น ที่นิยมใช้กันก็คือ วิธีที่ 2 ในเทอร์โบซีได้เตรียมฟังก์ชันในการฝังโปรแกรมเอาไว้ คือ

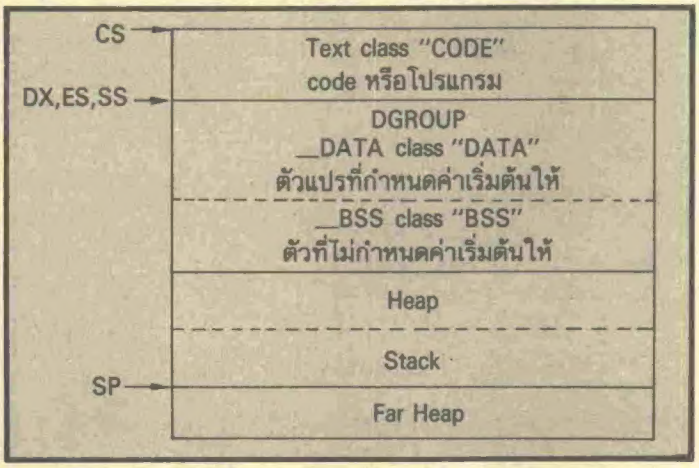
```
void keep (int status, int size);
```

ส่วนของไมโครซอฟต์ซี คือ

```
void_dos_keep (unsigned retcode, unsigned
memsize);
```

ซึ่งทั้งสองฟังก์ชันเป็นฟังก์ชันที่เรียกใช้ฟังก์ชัน 31H ของดอส

โดยที่ status และ retcode ก็คือ Exit Code ที่จะส่งออกไปยังดอสหลังทำงานโปรแกรมเสร็จ ส่วน size และ memsize คือ ขนาดของโปรแกรมที่จะฝังในหน่วยความจำ การคำนวณขนาดของโปรแกรมที่ทำได้ไม่ยากนัก สำหรับเทอร์โบซี ซึ่งในตัวอย่างโปรแกรม จะเป็นโปรแกรมในโมเดลแบบ small ส่วนของฮีพ (heap) จะอยู่ก่อนสแต็ก ดังนั้นการคำนวณจึงไม่ต้องคิดถึงฮีพ เพราะฮีพจะถูกรวมเข้าไปอัตโนมัติ แต่ต้องกำหนดไว้ให้เพียงพอเท่านั้น ถ้าเป็นโมเดลอื่นซึ่งฮีพจะอยู่ต่อจากสแต็ก ถ้าเราใช้งานฮีพในโปรแกรมเรสซิเดนต์ก็ต้องนำฮีพมาใช้ด้วย การคำนวณขนาดโปรแกรมก็จะต้องนำมาติดด้วย ดังแสดงในโปรแกรมตัวอย่างของไมโครซอฟต์ซีได้ในรูปที่ 7



รูปที่ 7 แสดงการจัดเรียงหน่วยความจำสำหรับโปรแกรมของเทอร์โบซีโมเดล small

การคำนวณขนาดโปรแกรมของเทอร์โบซี

ในโปรแกรมของเทอร์โบซีนั้น เราสามารถจะกำหนดขนาดของสแต็กและฮีพได้โดยตรงทางตัวแปรของตัวแปลภาษา คือ

```
extern unsigned __stklen = 512
extern unsigned __heaplen = 128
```

การคำนวณขนาดโปรแกรมดูตามรูปที่ 7 วิธีคำนวณก็คือ หาตำแหน่งของสแต็กแล้วลบออกด้วยตำแหน่ง Program Segment Prefix (PSP) ซึ่งรู้ค่าได้จากตัวแปร _psp ของเทอร์โบซี

```
segread (&seg);
myss = seg.ss;
stack = __sp;
sizeprogram = myss + ((stack + 50)/16)- __psp;
```

การคำนวณขนาดโปรแกรมของไมโครซอฟต์ซี

โปรแกรมของไมโครซอฟต์ซี การกำหนดขนาดของสแต็กกำหนดตอนเลือก Option ของ Compiler และการคำนวณขนาดโปรแกรมนั้น จะต้องรวมฮีพเข้าไปด้วย เนื่องจากการจัดเรียงหน่วยความจำของโปรแกรมที่เขียนด้วยไมโครซอฟต์ซี ฮีพจะอยู่ต่อจากสแต็ก

```
segread (&seg);
myss = seg.ss;
stack = getsp();
sizeprogram = heaplen/16 + (myss + (stack + 50)-
              /16) _psp;
```

โดย heaplen คือตัวแปรที่เรากำหนดขึ้นมาเอง ค่าที่กำหนดให้ตามความต้องการของโปรแกรมเรสซิเดนต์ที่ใช้งาน

การเรียกใช้งานโปรแกรมเรสซิเดนต์

หลังจากฝังโปรแกรมลงในหน่วยความจำแล้ว การเรียกใช้งานโปรแกรมเรสซิเดนต์จะถูกเรียกใช้ผ่านทางอินเตอร์รัพต์ ดังนั้นโปรแกรมเรสซิเดนต์ส่วนใหญ่จึงต้องทำการเปลี่ยนแปลงค่าในตารางอินเตอร์รัพต์เวกเตอร์ ให้ชี้มาที่ฟังก์ชันที่จะให้ทำงานแทน เมื่อเกิดอินเตอร์รัพต์ เช่น โปรแกรมนาฬิกา ซึ่งจะแสดงเวลาทุก ๆ วินาทีเมื่อเวลาเดินก็จะเปลี่ยนค่าในตารางอินเตอร์รัพต์หมายเลข 1C หรือโปรแกรม SideKick ซึ่งเรียกใช้งานโปรแกรมทางคีย์บอร์ดก็จะเปลี่ยนอินเตอร์รัพต์หมายเลข 9 เป็นต้น

ในเทอร์โบซีจะใช้ฟังก์ชันในการกำหนดหรืออ่านค่าอินเตอร์รัพต์เวกเตอร์

```
void setvect (int intr__num, void interrupt (*isr)( ));
void interrupt (*getvect (int intr__num))();
```

ในไมโครซอฟต์ซีจะใช้

```
void __dos__ setvect (unsegned intnum, interrupt
for *handler);
void (interrupt for *__dos__ getvect (intnum))();
```

ข้อจำกัดของดอส (Non-rectrant)

ข้อจำกัดของดอสที่สำคัญอันหนึ่งก็คือ ฟังก์ชันบริการต่าง ๆ ของดอสนั้นไม่สามารถจะเรียกซ้ำซ้อนได้ (Non-rectrant) เนื่องจากดอสจะสร้างตัวแปรและกำหนดใช้สแต็กในหน่วยความจำที่แน่นอนตำแหน่งหนึ่ง การเรียกใช้ฟังก์ชันของดอสซ้ำซ้อนก่อนที่ดอสจะทำงานของฟังก์ชันนั้นเสร็จ ก็จะทำให้เกิดการทำลายข้อมูลและสแต็กของเดิมได้ ซึ่งทำให้เครื่องหยุดการทำงานได้ (แต่ก็มีวิธีการแก้ไข เพื่อที่จะสามารถเรียกใช้ฟังก์ชันของดอสบางอย่างได้โดยปลอดภัย ซึ่งจะกล่าวถึงในตอนต่อไป) ดังนั้นในฟังก์ชันที่เป็นอินเตอร์รัพต์รูทีนในตัวอย่างโปรแกรมจึงไม่สามารถใช้ฟังก์ชัน printf, puts ฯลฯ ได้

สำหรับเทอร์โบซีตั้งแต่เวอร์ชัน 1.5 ขึ้นไป ได้สร้างฟังก์ชันการแสดงผลบนจอ ซึ่งไม่ผ่านดอสแต่ผ่านทางไบออสหรือเขียนลงไปตรง ๆ ที่หน่วยความจำสำหรับการแสดงผลเลย ซึ่งได้แก่พวก cprintf, cputs ฯลฯ ซึ่งสามารถนำไปใช้งานได้ในฟังก์ชันที่เป็นอินเตอร์รัพต์รูทีนได้

ส่วนของไมโครซอฟต์ซี จะต้องสร้างฟังก์ชันการแสดงผลขึ้นมาเอง เป็นการเขียนที่หน่วยความจำแสดงผลโดยตรงเป็นภาษาแอสเซมบลี

รูปที่ 8 แสดงโปรแกรม CLOCK.C สำหรับเทอร์โบซี

```
#include <dos.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <graphics.h>

#define SCREEN_SEG       0xb000
#define SCREEN_OFF       0x0840
#define INTENSITY        0x7f
#define NORMAL           0x77
#define BLINK_NORMAL     0xf7
#define BLINK_INTENSITY  0xff
#define UNDERLINE        0x01
#define REVERSE          0x70

/*****************************************
 *   Variable use in resident process    *
 *****************************************/
extern unsigned _stklen  = 512;
extern unsigned _heaplen = 128;
static unsigned sizeprogram;
static unsigned myss;
static unsigned stack;

/***************************
 * Timer Vector Interrupt  *
 ***************************/
#define TIMER   0x1c

/*******************************
 * Oldtimer interrupt address  *
 *******************************/
static void interrupt (*oldtimer)();

/*******************************
 * Newtimer interrupt address  *
 *******************************/
static void interrupt newtimer();

/*****************************
 * Function to display time  *
 *****************************/
static void timer ();

/*****************************************
 *  Global Variable use to count timer trige *
 *****************************************/
static unsigned char  counter1;
static unsigned char  counter2;
static unsigned char  hour;
static unsigned char  min;
static unsigned char  sec;
static unsigned char  row=1;
static unsigned char  column=73;

void resinit()
{
        struct SREGS seg;

        segread(&seg);
        myss = seg.ss;
        stack = _SP;
        oldtimer = getvect(TIMER);
        setvect(TIMER, newtimer);
        sizeprogram = (unsigned)myss +
          ((((unsigned)stack + 50) / 16) -
          (unsigned) _psp;
        keep(0, sizeprogram);
}

void interrupt newtimer()
{
        unsigned intsp,intss;

        (*oldtimer)();
        intsp = _SP;
        intss = _SS;
        _SP = stack;
        _SS = myss;
        enable();
        timer ();
        disable();
        _SP = intsp;
        _SS = intss;
        enable();
}
```

→

รูปที่ 8 (ต่อ)

```
void updatetime ()
{
        int oldx,oldy;

       if (sec == 59)
          {
          sec = 0;
          if (min == 59)
             {
             min = 0;
             if (hour == 23) hour = 0;
             else hour++;
             }
          else min++;
          }
       else sec++;
       oldx = wherex ();
       oldy = wherey ();
       gotoxy (column,row);
       cprintf("%02d:%02d:%02d",hour,min,sec);
       gotoxy (oldx,oldy);
}
void timer ()
{
       counter1++;
       if (counter2 !=0)
          {
          if (counter1==18)
             {
             updatetime();
             counter1 = 0;
             if (counter2 ==4)
                counter2 = 0;
             else
                counter2++;
             }
          }
       else
             if (counter1 == 19)
                {
                updatetime();
                counter1 = 0;
                counter2++;
                }
}

void readtime()
{
        struct time timep;

        counter1 = counter2 =0;
        gettime (&timep);
        hour = timep.ti_hour;
        min  = timep.ti_min;
        sec  = timep.ti_sec;
}

main (int argc,char *argv[])
{
   if (argc == 3)
      {
      column = atoi(argv[1]);
      row = atoi(argv[2]);
      }
   puts ("Clock was modify interrupt vector
         No. 1C,");
   puts ("And terminated-stay resident in
         memory.");
   readtime ();
   textattr (INTENSITY);
   resinit ();
}
```

รูปที่ 9 แสดงโปรแกรม CLOCK.C สำหรับไมโครซอฟต์ซี

```
#include <dos.h>
#include <time.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <memory.h>
#include "video.h"

extern unsigned getsp (void);   /* Assambly
   routine get stack pointer */
extern void do_resident (void); /* Assambly
   routine done resident function */

/*************************************
 *   Variable use in resident process  *
 *************************************/
extern unsigned _psp;
unsigned heaplen = 128;
unsigned sizeprogram;
unsigned myss;
unsigned stack;

/***************************
 * Timer Vector Interrupt *
 ***************************/
#define TIMER   0x1c

/*******************************
 * Oldtimer interrupt address *
 *******************************/
void (interrupt far *oldtimer)();

/*******************************
 * Newtimer interrupt address *
 *******************************/
void interrupt far newtimer();

/*****************************
 * Function to display time *
 *****************************/
void timer ();
```

รูปที่ 9 (ต่อ)

```
/*********************************************
 *  Global Variable use to count timer trige *
 *********************************************/
static unsigned char  counter1;
static unsigned char  counter2;
static unsigned char  hour;
static unsigned char  min;
static unsigned char  sec;
static unsigned char  row=1;
static unsigned char  column=73;

/* Initialize program and terminate
   stay-resident  */
void resident_init ()
{
   struct SREGS seg;

   oldtimer = _dos_getvect(TIMER);
   _dos_setvect(TIMER, newtimer);
   segread (&seg);
   myss = seg.ss;
   stack = getsp ();
   sizeprogram = myss + (stack + 50)/16 +
      heaplen/16 - _psp;
   _dos_keep(0, sizeprogram);
}

void interrupt far newtimer()
{
   unsigned intsp,intss;

        (*oldtimer) ();
        do_resident ();
}

void printtime ()
{
        char str[8];

   f_gotoxy (column,row);
        itoa (hour,str,10);
        if (hour < 10)
                f_putc ('0');
        f_puts (str);
        f_putc (':');
        itoa (min,str,10);
        if (min < 10)
                f_putc ('0');
        f_puts (str);
        f_putc (':');
        itoa (sec,str,10);
        if (sec < 10)
                f_putc ('0');
        f_puts (str);
}

void updatetime ()
{
   if (sec == 59)
      {
           sec = 0;
           if (min == 59)
                {
                min = 0;
                if (hour == 23) hour = 0;
                else hour++;
                }
           else min++;
           }
   else sec++;
        printtime ();
}

void timer ()
{
   counter1++;
   if (counter2 !=0)
      {
      if (counter1==18)
                {
                updatetime();
                counter1 = 0;
                if (counter2 ==4)
                   counter2 = 0;
                else
                   counter2++;
                }
           }
   else
           if (counter1 == 19)
                {
                updatetime();
                counter1 = 0;
                counter2++;
                }
}

void readtime()
{
   struct dostime_t time;

   counter1 = counter2 =0;
   _dos_gettime (&time);
   hour = time.hour;
   min  = time.minute;
   sec  = time.second;
}

main (int argc,char *argv[])
{
   if (argc == 3)
      {
      column = atoi(argv[1]);
      row = atoi(argv[2]);
      }
        fast_display ();
   puts ("Clock was modify interrupt vector
         No. 1C,");
   puts ("And terminated-stay resident in
         memory.");
   readtime ();
   textattr (HighDisplay);
   resident_init ();
}
```

รูปที่ 10 แสดง TSRASM.ASM ฟังก์ชันการเขียนโปรแกรมอินเตอร์รัพต์

```
      dosseg
      .model small
      .data
      extrn _myss  : word
      extrn _stack : word
intrss   dw   ?
intrsp   dw   ?

      .code
extrn  _timer : near
public _getsp

;-------------------------
; return the stack pointer
; stackpointer = getsp ()
;-------------------------
_getsp proc near
       mov  ax,sp
       ret
_getsp endp
public _do_resident
;-----------------------------------------
; swap the stack and call the popup routine
;-----------------------------------------
_do_resident proc near
      cli               ; dispable interrupt
      mov  intrss,ss    ; save the current stack
      mov  intrsp,sp
      mov  ss,_myss     ; set TSR's stack
      mov  sp,_stack
      sti               ; enable interrupt
      call _timer       ; pop up the TSR routine
      cli               ; disable interrupt
      mov  ss,intrss    ; restore current stack
      mov  sp,intrsp
      sti
      ret
_do_resident endp

end
```

รูปที่ 11 แสดง VIDEO.H เฮดเดอร์ไฟล์สำหรับฟังก์ชันการแสดงผล

```
#define NoDisplay       0x00
#define LowDisplay      0x07
#define HighDisplay     0x0f
#define UnderLineLow    0x01
#define UnderLineHihg   0x09
#define ReverseLow      0x70
#define ReverseHigh     0x7f
#define BlinkLow        0x87
#define BlinkHigh       0x8f
#define UnderBlinkLow   0x81
#define UnderBlinkHigh  0x89
#define RevBlinkLow     0xf0
#define RevBlinkHigh    0xf8
#define True            01
#define False           0

/* Control and status port address */
#define Mono_crt_status   0x3ba
#define Color_crt_status  0x3da
#define Mono_Seg     0xb000
#define Color_Seg    0xb800

typedef struct
{
    unsigned cursor_pos;        /* current possition on screen */
    unsigned stport;            /* CRT I/O port address        */
    unsigned char  char_attrib; /* cureen screen atrribute     */
}   SCREEN_DATA;

/*-----------------------------------------------------------------*/
void            fast_display ();
void            f_gotoxy (char x,char y);
void            f_putc (unsigned char c);
void            f_puts (unsigned char  *str);
void            textattr (unsigned char attrib);
void            set_size_cursor (unsigned char top,unsigned char bottom);
void            cursor_on (void);
void            cursor_off (void);
```

รูปที่ 12 แสดงฟังก์ชันการแสดงผลด้วยภาษาซี Videoc.c

```
#include <stdio.h>
#include <string.h>
#include <memory.h>
#include <dos.h>
#include <bios.h>
#include <ctype.h>
#include "video.h"

extern void     fast_putc (char);
extern void     fast_puta (char);
extern int      fast_getc (void);

extern SCREEN_DATA   direct_video;
unsigned int  screenseg;

void fast_display ()
{
   union REGS  regs;

   int86(0x11,&regs,&regs);
   if((regs.x.ax & 0x30)==0x30)
                {
      screenseg=Mono_Seg;
      direct_video.stport=Mono_crt_status;
                }
   else
                {
      screenseg=Color_Seg;
      direct_video.stport=Color_crt_status;
                }
   direct_video.cursor_pos=0;
   direct_video.char_attrib=LowDisplay;
}

void f_gotoxy(char x,char y)
{
   direct_video.cursor_pos=((int)(y-1) * 80) +(int) (x-1) << 1;
}

void set_size_cursor(unsigned char top,unsigned char bottom)
{
  unsigned  int  far *cursormode = (unsigned int far *) 0x004000601;
  unsigned  int  far *videoport  = (unsigned int far *) 0x004000631;

  if(!screenseg==Mono_Seg) {
    top=top-4;
    bottom=bottom-4;
  }
  outp(*videoport,10);
  outp(*videoport+1,top);
  outp(*videoport,11);
  outp(*videoport+1,bottom);
  *cursormode=(top << 8) | bottom;
}

void cursor_on()
{
  unsigned  int  far *cursormode = (unsigned int far *) (0x004000601);
  unsigned  int  far *videoport  = (unsigned int far *) (0x004000631);

  outp(*videoport,10);
  outp(*videoport+1,((unsigned char )(*cursormode >> 8)) | 0x20 );
  outp(*videoport,11);
```

รูปที่ 12 (ต่อ)

```
  outp(*videoport+1,(unsigned char )*cursormode);
}

void textattr (unsigned char attr)
{
  direct_video.char_attrib=attr;
}

void f_puta (unsigned char attr)
{
        fast_puta (attr);
   direct_video.cursor_pos=direct_video.cursor_pos+2;
   if(direct_video.cursor_pos > 3998)
      direct_video.cursor_pos =0;
}

void f_putna (int n ,unsigned char attr)
{
  int i;

  for (i=0;i<n;i++) {
    f_puta(attr);
  }
}

void f_putc (unsigned char c)
{
        fast_putc(c);
   direct_video.cursor_pos=direct_video.cursor_pos+2;
   if(direct_video.cursor_pos > 3998)
      direct_video.cursor_pos =0;
}

void f_puts(unsigned char  *str)
{

  while(*str) {
     switch(*str) {
        case '\r' :
        case '\n' : break;
        case '\b' : direct_video.cursor_pos=direct_video.cursor_pos-2;
                    break;
        default   : f_putc(*str);
                    break;
     }
     str++;
  }
}
```

รูปที่ 13 แสดงฟังก์ชันการแสดงผลด้วยภาษาแอสเซมบลี videoa.asm

```
      dosseg
      .model small

SCREEN_DATA struc
   cursorpos dw    0       ; cursor position
   stport    dw    0       ; CRT I/O port
   attribute db    0       ; attribute of char
SCREEN_DATA ends

      .data

      public _direct_video
      _direct_video  SCREEN_DATA <>
      extrn  _screenseg     : word

      .code

;-----------------------------
; fast put character to screen
; fast_putc (char c);
;-----------------------------
```

รูปที่ 13 (ต่อ)

```
        public _fast_putc
_fast_putc    proc near
        push    bp
        mov     bp,sp
        push    es
        push    di
        mov     es,_screenseg
        mov     di,_direct_video.cursorpos
        mov     dx,word ptr _direct_video.stport
        mov     ch,byte ptr _direct_video.
                attribute
        mov     cl,byte ptr [bp+4]
putc0:
        cmp     dx,03bah
        jz      monoc
        in      al,dx
        test    al,1
        jnz     putc0
putc1:
        in      al,dx
        test    al,1
        jz      putc1
monoc:
        mov     ax,cx
        stosw
        pop     di
        pop     es
        pop     bp
        ret
_fast_putc endp

;-----------------------------------------
; fast put attribute of character to screen
; fast_puta (unsigned char attribute);
;-----------------------------------------
        public _fast_puta
_fast_puta    proc near
        push    bp
        mov     bp,sp
        push    es
        push    di
        mov     es,_screenseg
        mov     di,_direct_video.cursorpos
        mov     dx,word ptr _direct_video.stport
        mov     cl,byte ptr [bp+4]
        inc     di
puta0:
        cmp     dx,03bah
        jz      monoa
        in      al,dx
        test    al,1
        jnz     puta0
puta1:
        in      al,dx
        test    al,1
        jz      puta1
monoa:
        mov     al,cl
        stosb
        pop     di
        pop     es
        pop     bp
        ret
_fast_puta endp

;-----------------------------------------
;get character and attribute from screen
;byte high is an chararter code
;byte low is  an attribute
;int fast_getc (void)
;-----------------------------------------
        public _fast_getc
_fast_getc    proc near
        push    ds
        push    si
        mov     cx,_screenseg
        mov     si,_direct_video.cursorpos
        mov     dx,word ptr _direct_video.stport
        mov     ds,cx
getc0:
        cmp     dx,03bah
        jz      monog
        in      al,dx
        test    al,1
        jnz     getc0
getc1:
        in      al,dx
        test    al,1
        jz      getc1
monog:
        lodsw
        pop     si
        pop     ds
        ret
_fast_getc endp

end
```

■

โปรแกรมตัวอย่าง

โปรแกรมตัวอย่างเป็นโปรแกรมนาฬิกาแสดงเวลาตามตำแหน่งที่สามารถกำหนดได้ตอนเรียกโปรแกรมพิมพ์อยู่ ในรูปที่ 8 เขียนด้วยเทอร์โบซี ส่วนรูปที่ 9, 10, 11 และ 12 เขียนด้วยไมโครซอฟต์ซี โปรแกรมจะถูกแบ่งออกเป็นสองส่วนด้วยกัน คือ

1. ส่วนการติดตั้งโปรแกรม

ในส่วนนี้เป็นการกำหนดค่าเริ่มต้นให้ตัวแปรต่าง ๆ ที่จะใช้งาน การเปลี่ยนแปลงค่าในตารางอินเตอร์รัพต์เวกเตอร์ คำนวณหาขนาดของโปรแกรมที่ฝังลงในหน่วยความจำหลังจากที่ฝังโปรแกรมและเลิกออกจากโปรแกรม

2. ส่วนบริการอินเตอร์รัพต์

ในส่วนนี้ก็คือ ฟังก์ชันที่เป็นโปรแกรมอินเตอร์รัพต์แทนอินเตอร์รัพต์เดิม ในฟังก์ชัน newtimer( ) ซึ่งเป็นฟังก์ชันที่จะมีการเรียกอินเตอร์รัพต์เดิมก่อน เพื่อให้ระบบของเครื่องทำงานได้ถูกต้อง จากนั้นจึงเรียกฟังก์ชันที่เราเพิ่มเติมลงไป ในเทอร์โบซีเราสามารถจะทำการเปลี่ยนสแต็กได้โดยตรง แต่ในไมโครซอฟต์ซีทำไม่ได้ ดังนั้นส่วนนี้ของโปรแกรมไมโครซอฟต์ซีจึงต้องเขียนด้วยภาษาแอสเซมบลี การเปลี่ยนสแต็กมาใช้สแต็กของที่กำหนดก็เพื่อป้องกันไม่ให้เกิดการเขียนทับข้อมูลส่วนอื่นของระบบ เพราะขณะเกิดอินเตอร์รัพต์ขึ้นมานั้น สแต็กชี้อยู่ที่ใดไม่สามารถรู้ได้ ถ้าในฟังก์ชันของเรามีการเรียกฟังก์ชันย่อยหรือมีการผ่านค่าระหว่างการเรียกฟังก์ชันก็อาจจะทำให้สแต็กล้น แล้วไปเขียนทับข้อมูลส่วนใดของระบบก็ได้

การใช้งานโปรแกรมตัวอย่าง

โปรแกรมตัวอย่างของเทอร์โบซี ในรูปที่ 8 การใช้งานคอมพายล์ได้ใช้ TC.EXE หรือ TCC.EXE ก็ได้ โดยเลือกโมเดลให้เป็น small โดยเรียก

การเรียกโปรแกรมทำได้โดย

```
C> TCC -ms clock.c
C> Clock 1 72
```

↑ บอกตำแหน่งคอลัมน์
↑ บอกตำแหน่งแถว

สำหรับของไมโครซอฟต์ซี ในรูปที่ 9-13 ส่วนที่เป็นภาษาซี คือ รูปที่ 9 ส่วนที่เป็นภาษาแอสเซมบลี คือ รูปที่ 10 ส่วนรูปที่ 11, 12 และ 13 นั้นเป็นฟังก์ชันการแสดงผล ซึ่งเขียนลงในหน่วยความจำแสดงผลโดยตรง

การคอมพายล์โปรแกรมที่เขียนด้วยภาษาแอสเซมบลีจะต้องใช้ masm เวอร์ชัน 5.0 โดยเรียก

```
C > masm-me tsrasm ;
C > masm-me videoa ;
```

หลังจากนั้นจะได้ .obj นำมาร่วมกับภาษาซีอีกครั้งหนึ่งโดยเรียก

```
C > CL-AS CLOCK.C tsrasm videoc.c videoa
```

วิธีการเรียกใช้เหมือนกับของเทอร์โบซี

ก่อนจบตอน

สำหรับเทคนิคโปรแกรมในฉบับนี้คงจะต้องหยุดลงเพียงแค่นี้ก่อน ในฉบับหน้าเราจะนำความรู้ที่ได้จากฉบับนี้ไปใช้งานต่อและเพิ่มเทคนิคต่าง ๆ ในการเขียนโปรแกรมเรสซิเดนต์ที่สามารถเรียกใช้งานได้ทันทีทันใดโดยกด hot key ก่อนจะจบก็ขอฝากท่านผู้อ่าน ท่านที่มีข้อเสนอแนะหรือติชม มุมภาษาซียินดีที่จะปรับปรุงเนื้อหาให้ตรงตามความต้องการของท่านผู้อ่านเสมอ...